REMARKS

Claims 1-20 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Alleged Obviousness, Claims 1, 2 and 4

The Office Action rejects claims 1, 2 and 4 under 35 U.S.C. § 103(a) as being unpatentable over Cohen et al. (U.S. Patent No. 6,389,462 B1) in view of Kayashima (U.S. Patent No. 6,195,366 B1) and further in view of Jade et al. (U.S. Patent No. 5,944,823). This rejection is respectfully traversed.

As to claim 1, the Office Action states:

> In reference to claim 1, Cohen teaches a method in a network device for caching Hyper Text Transfer Protocol (HTTP) data transported in an Internet Protocol (IP) Datagram sent on a socks connection established over a Transmission Control Protocol (TCP) connection between a source port on a source device and destination port on a destination device (Cohen, Abstract and column 6 lines 21-46; Cohen discloses caching HTTP connection data transported in an IP packet over TCP between source and destination), said method comprising the steps of:
>
> Identifying elements of an incoming IP Datagram, comprising:
> the source device,
> the destination device,
> the port on the source device,
> the port on the destination device,
> (Cohen, column 6, lines 21-67; Cohen discloses identifying source and destination device and ports from a packet);
>
> determining whether the incoming IP Datagram originates from a socks client or from a socks server (Cohen, column 7 lines 1-35; Cohen discloses determining if packet is originated by client or origin server):
>
> in response to the incoming IP Datagram originates from a socks client (Cohen, column 6 lines 21-67; Cohen discloses IP packets/requests from client):
>
> terminating the TCP connection and the socks connection (Cohen, column 6 line 21 – column 7 line 67; Cohen discloses terminating TCP connection by interrupting transmission between client and origin server);
>
> Cohen fails to teach identifying the socks connection in a table. However; Kayashima discloses socks connection, terminating socks

connection, and identifying the connections in tables (Kayashima, column 1 line 5 – column 2 line 36, column 5 line 42 – column 6 line 23 and figure 3);

It would have been obvious for one ordinary skilled in the art to identify the socks connection in a table as per the teachings of Kayashima for keeping track of the connection parameters.

Cohen also fails to teach identifying the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol. However, Jade discloses maintaining a connection table within an intermediary network device, the table contains protocol entries (Jade, column 1 lines 50-67 and column 4 lines 40-67);

It would have been obvious for one ordinarily skilled in the art to identify the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol as per the teachings of Jade for identifying the connection information protocol for data transmission.

determining whether said application level protocol is HTTP or not:

In response to said application level protocol being HTTP:

determining whether HTTP data requested by the incoming IP Datagram resides in a local cache within the network device (Cohen, Abstract, column 6 line 23 – column 8 line 52; Cohen discloses determining if HTTP data request packet is in a local proxy cache):

In response to the HTTP data requested by the incoming IP Datagram residing in a local cache:

building an outgoing IP Datagram comprising requested HTTP data retrieved from the local cache; and

sending said outgoing IP Datagram to the socks client originator of the incoming IP Datagram (Cohen, Summary, column 6 line 23 – column 8 line 52; Cohen discloses building an IP packet comprising the HTTP data request and sending it to the client).

Office Action dated September 20, 2004, pages 2-4.

Claim 1, which is representative of the other rejected independent claims 10 and 11 with regard to similarly recited subject matter, reads as follows:

1. A method in a network device for caching Hyper Text Transfer Protocol (HTTP) data transported in an Internet Protocol (IP) Datagram sent on a socks connection established over a Transmission Control Protocol (TCP) connection between a source port on a source device and a destination port on a destination device, said method comprising the steps of:

identifying elements of an incoming IP Datagram, comprising:

the source device;

the destination device;

the port on the source device; and
the port on the destination device;
determining whether the incoming IP Datagram originates from a socks client or from a socks server;
in response to the incoming IP Datagram originating from a socks client:
terminating the TCP connection and the socks connection;
identifying the socks connection in a table;
identifying the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol; and
determining whether said application level protocol is HTTP or not;
in response to said application level protocol being HTTP:
determining whether HTTP data requested by the incoming IP Datagram resides in a local cache within the network device; and
in response to the HTTP data requested by the incoming IP Datagram residing in a local cache:
building an outgoing IP Datagram comprising requested HTTP data retrieved from the local cache; and
sending said outgoing IP Datagram to the socks client originator of the incoming IP Datagram.

Cohen, Kayashima and Jade, taken alone or in combination, fail to teach or suggest determining whether the incoming IP Datagram originates from a socks client or from a socks server.

Cohen is directed to a system that transparently redirects an HTTP connection request that is directed to an origin server to a proxy cache. The redirection is performed by a proxy redirector that translates the destination address of packets directed to the origin server to the address of the proxy. During a handshaking procedure, a TCP connection is transparently established between the client and the proxy cache. When the client transmits a GET request to what it thinks is the origin server, a request specifies the complete address of an object at that origin server that it wants a copy of, the proxy redirector modifies the complete address specified in that GET request before it is sent to the proxy cache.

Thus, Cohen merely performs a redirection of a connection from an origin server to a proxy server. Cohen does not teach or suggest determining whether the incoming IP

Datagram originates from a socks client or from a socks server. The Office Action alleges that Cohen teaches this feature at column 7, lines 1-35, which reads as follows:

> With the IP address of the origin server determined and returned to the client, the browser establishes a TCP connection between the client and the origin server through a three-way handshaking process. Specifically, a SYN packet, addressed to the IP address of the selected origin server, is sent by the client. Handshaking is completed when the client receives an acknowledgement of receipt of that SYN packet through an ACK SYN packet sent by that origin server, and responds with a ACK packet to the origin server. The browser then sends a GET request that specifies the particular requested object.
>
> In accordance with the present invention, once the IP address of the origin server corresponding to the logical URL name is determined through the DNS look-up, proxy redirector 104, rather than establishing a TCP connection to the origin server at the determined IP address, transparently establishes a TCP connection between the client and a proxy. If the requested object is stored in the cache, a copy of that object is transparently returned to the requesting client. A TCP connection, therefore, is not established over the Internet 105, to the actual origin server 107 to provide the requested object to the requesting client. The coss of transmitting the request to the origin server over the Internet and transmitting the copy of the requested object back over the Internet are thereby saved in addition to the time required for transmitting such a request over the Internet and waiting for a response from the origin server. If the proxy cache to which the request is directed does not contain the requested object, a separate TCP connection is established between the proxy cache and the origin server to obtain a copy of the requested object. When the proxy cache then receives the copy of the requested object from an origin server over that separate TCP connection, the copy is forwarded to the client over the original TCP connection that was established between the client and the proxy cache.

In this section, Cohen describes determining the IP address of the origin server and, rather than establishing a connection to the origin server, a connection is established to a proxy server. The determination of the IP origin server is performed by reading the packets in the request which is sent from the client which is described in column 6, lines 21-67, which read as follows:

> With reference to FIG. 1, a plurality of clients 101-1-101-N are connected to a local area network (LAN) 102, such as an Ethernet. LAN 102, which, in turn, is connected through a router 103 to a Level 4 (L4) switch 104 (proxy redirector) which interfaces the LAN with a wide area network (WAN) 105, such as the Internet. Although shown as two separate elements, the functionalities of router 103 and proxy redirector

> 104 can in actual practice be combined in a single unit. All requests from any of the clients connected to LAN 102 for objects stored in servers connected to the Internet 105 traverse proxy redirector 104 onto the Internet. The packets comprising such requests, which include the standardized packets that establish a TCP connection, are directed to an IP destination address and port number indicated in the IP header of each packet originating from a client source address that includes a client IP address and port number. Similarly, responses to such requests from an origin server connected to Internet 105 are directed via an IP destination address that is equal to the client's IP address and port number from which the request originated, and have as a source address the server's IP address and port number. All such packets directed to any of the clients 101-1-101-N from any server connected to Internet 105 pass through proxy redirector 104.
>
> When any of the clients connected to LAN 102, such as client 101-1, makes a request through a browser for an object by specifying a logical URL, a domain name server (DNS) 106 connected locally or on Internet 105, as shown, is accessed to perform a database look-up based on that logical name. An associated IP address is then returned to the browser. The IP address returned to the browser is the IP address of a particular origin server which contains the 5 object requested through the logical URL. Since a logical name may in fact be associated with a plurality of essentially equivalent origin servers, such as servers 107 and 109, the particular IP address returned to the client browser chosen by DNS 106 may be determined in a round-robin manner. When DNS 106 selects an origin server corresponding to the logical URL, the IP address of the selected origin server, such as, for example, the IP address of origin server 107, is returned to the browser in the requesting client 101-1. That IP address then serves as the IP address to which packets directed to the origin server from the client are directed. Conventionally, http requests are usually directed to port 80 of an origin server.

As clearly described in this section of Cohen, the packets comprising the request from the client contain an IP destination address and port number in addition to the clients IP address and port number. There is nothing in either of these sections, or any other section of Cohen, that teaches or suggests determining whether the incoming IP Datagram originates from a socks client or from a socks server as all requests in the Cohen reference originate from the client and, instead of a connection being established to the origin server, a connection is established to a proxy server.

Additionally, Applicant is not merely claiming determining whether the incoming IP Datagram originates from a client or from a server, which Cohen does not teach or suggest, but determining whether the incoming IP Datagram originates from a socks client or from a socks server. A Socks protocol is not mentioned anywhere in the Cohen reference. Moreover, the Kayashima and Jade references which are used in combination with the Cohen reference to teach all of the features of these claims, fail to make up for the deficiencies of the Cohen reference.

Kayashima is directed to a system that conducts a connectionless communication in a network communication system including a client, a server, and a plurality of proxy servers which are disposed on a transmission path between the client and server. While Kayashima may mention a Socks protocol, Kayashima does not determine whether the incoming IP Datagram originates from a socks client or from a socks server.

Jade is directed to a system where a firewall isolates computer and network resources inside the firewall from networks, computers and computer applications outside the firewall. Jade fails to teach or suggest determining whether the incoming IP Datagram originates from a socks client or from a socks server.

Additionally, the combination of Cohen, Kayashima, and Jade fail to teach or suggest in response to the incoming IP Datagram originating from a socks client, terminating the TCP connection and the socks connection. The Office Action alleges that Cohen teaches this feature in column 6, line 21 to column 7, lines 67, most of which is shown above and the remaining portion reads as follows:

> In the embodiment shown in FIG. 1, a proxy cache 110-1 is illustratively shown connected to a LAN 111, which is connected to LAN 102 through a router 112. Another proxy cache 115 is shown connected on a different LAN 116 through router 103. Other proxy caches can be located anywhere on LANs 102, 111, or 116, on another LAN connected to the Internet 105 such as proxy cache 117. Proxy redirector 104 selects one of the available proxy caches to which to forward client requests based on a metric such as least-loaded or round-robin, based on IP header information such as the origin server IP address. With respect to the latter, all objects from a specific origin server will be served by a specific proxy.
>
> In the preferred embodiment described herein, proxy redirector 104 includes a programmable network element of the type described in co-pending U.S. patent application Ser. No. 09/190,355, filed Nov. 12, 1998, which application is incorporated herein by reference. As described in that application, that programmable network element in the preferred embodiment runs on a Linux machine. As shown in FIG. 2, the programmable network element 200 includes a dispatcher process 202 with which plural different gateway programs (204, 205 and 206) register and request access to IP packets that fit specific descriptions. Such

> programs are loaded through an admission daemon 210 via a local program injector 211 or a remote program injector 212. A gateway program, for example, can request access to incoming packets to network interface 208 that match certain source and destination IP address ranges and port numbers. The dispatcher process 202 uses a packet filter 203 in the Linux kernel 201 to obtain packets requested by the gateway programs and uses a raw IP socket 215 to send packets that have been manipulated in accordance with the gateway program back to the kernel for output back to the network through filter 203 through network interfaces 208. Library functions are provided in the programmable network element that enable a gateway program to communicate with the dispatcher process 202 to register rules that specify the type of IP packets that a gateway program wants diverted to it. A gateway program can request either a complete IP packet or only the IP and TCP header of a packet and can change both the header and payload of a packet.

(Column 7, line 36 to column 8, line 10)

This rather lengthy section of Cohen merely describes determining the IP address from the client request of the origin server and, rather than establishing a connection to the origin server, establishing a connection to a proxy server. As discusses above, Cohen does not teach or suggest determining if the incoming request is from a client as all the requests in the Cohen system originate from the client and Cohen does not teach a Socks protocol. Additionally, Cohen does not teach or suggest terminating the TCP connection and the socks connection; instead, Cohen teaches, rather than establishing a connection from the client to the origin server, a connection is established from the client to a proxy server. Nowhere in any section of the Cohen reference is a connection ever taught to be terminated.

Furthermore, the Office Action alleges that Kayashima teaches identifying the socks connection in a table, in response to the incoming IP Datagram originating from a socks client. While Kayashima may teach a Socks connection table, the table in the Kayashima reference is used as a primary connection table to identify all connections and the Kayashima table is not identified in response to the incoming IP Datagram originating from a socks client.

Still further, the Office Action alleges that Jade teaches identifying the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol, in response to the incoming IP Datagram originating from a socks client. The Office Action alleges that this feature is taught by Jade at column 1, lines 50-67 and column 4, lines 40-67, which read as follows:

> In accordance with the invention, means are provided inside and outside a firewall for cooperatively producing tunneling effects, in response to certain types of requests initiated by objects outside the firewall, which effects result in creation of connections between such outside objects and resources inside the firewall. Connections so created have the unique property that they are effectively created from "inside out" as if they were requests originating from objects inside the firewall to destinations outside the firewall.
>
> The "types of requests" accommodated by such "tunneling" means are requests addressed to what are presently termed "trusted sockets". Trusted sockets are entries in a table of trusted sockets that is created and maintained exclusively inside the firewall. Each entry in that table includes the address of a "trusted" port, a protocol (e.g. a telecommunication protocol such as TCP/IP, NNTP, etc.) pre-associated with that address, and the identity of a host object inside the firewall (e.g. a host computer or a host application). Thus, it is understood that in order for an individual and/or object outside the firewall to initiate such a request that individual must be entrusted with the information that represents a trusted socket entry that is currently valid.

(Column 1, lines 50-67)

> The form of the trusted sockets table is illustrated in FIG. 4. Examples of 2 specific entries are shown at 30, and additional entries are implied at 31 by dotted lines extending downward from the second entry. Each entry consists of a port number, information defining a transmission protocol (usually, a burst packet transfer protocol), and information identifying a host object. The port number is an address inside the firewall assigned to the host object. As examples of protocols, the first two entries in the table list NNTP (Network News Transport Protocol) and HTTP (HyperText Transport Protocol).
>
> FIG. 5 shows in finer detail operations performed by the tunneling applications at interface servers A and B. Operations that are the same as operations shown in FIGS. 2 and 3 are identified by identical numerals. Operations that are parts of, or differ in some respect from, operations shown in FIGS. 2 and 3 are identified by the same numbers followed by letters (a, b, etc.). Other operations are identified by numbers different from those previously used.
>
> Operation 10a at server A, a composite of operations 10 and 12 of FIG. 2, is the creation and updating (expansion, modification. etc.) of the trusted sockets table and the copying of the latter to server B. Operation 11a at server A is the establishment or (as explained below) re-

> establishment of the control connection between (the tunneling applications at) servers A and B. A need to re-establish the control connection arises when the connection is unintentionally broken, and the operations required to detect and respond to such occurrences are shown at 46-48 in FIG. 5 (which are discussed further below).

(Column 4, lines 40-67)

In these sections, Jade describes a table of trusted sockets that have their configuration stored in a table and tunnels are produced using this trusted socket table. Nowhere in any section of Jade is an application level protocol associated with said socks connection referring to said table identified. Jade merely identifies the sockets to use for the connection from the trusted sockets in the trusted socket table. Additionally the sockets that are identified by the Jade system are not identified <u>in response to the incoming IP Datagram originating from a socks client</u>.

Even further, the Office Action alleges that Cohen teaches determining whether said application level protocol is HTTP or not in response to the incoming IP Datagram originating from a socks client. The Office Action alleges that this feature is taught at column 6, line 23 to column 8, line 52, most of which is shown above and the remained reads as follows:

> In the present invention, that programmable network element is operative in combination with a gateway program that manipulates the destination and source addresses of packets flowing there through in a manner to be described, as well as modifying, as will be described, information in the packet(s) containing the GET request that specifies the URL of the requested object. Specifically, the programmable network element in combination with the gateway program operates on packets associated with HTTP requests, which are determined from the destination port number. As previously noted, HTTP requests are conventionally addressed to port 80 of an origin server. Thus, the programmable network element/gateway program which together comprise proxy redirector 104 in this embodiment, captures through the dispatcher process of the programmable network element, packets directed to port 80 and then performs address translations on those captured packets to readdress these packets to a selected proxy. With respect to address translations, the gateway program translates the destination IP address of packets addressed to the origin server to the IP address of a selected proxy cache and translates the source IP address of such packets from that of the client to the IP address of proxy redirector 104. Further, in order for proxy redirector 104 to identify requests from plural client terminals that are directed to the same proxy, the source port number is translated to a bogus

> ghost port number at the proxy redirector. Thus, when proxy cache responds, the packets transmitted by the cache have a destination IP address of proxy redirector 104 at that bogus port number, which is distinctly associated with the client. The gateway program within proxy redirector 104 then translates the IP destination address of these responsive packets from the proxy to the IP address of the client and translates the bogus destination port number to the port number from which the client originated its request. Further, the gateway program translates the source IP address of such responsive packets from that of the proxy to the IP address of the origin server and the port number to the port (80) to which the client's requests were originally directed. Thus, the packets which are returned to the client from the proxy masquerade as if they had originated from the origin server to which the client "believed" its request had been sent.

In this rather lengthy section, Cohen describes determining which requests are HTTP request from the requests that are directed to port 80 of an origin server. The Cohen redirector captures the packets directed to port 80 and then performs address translations on those captured packets to readdress these packets to a selected proxy. Cohen does not determining whether said application level protocol is HTTP or not, but, rather, merely determines which requests are directed to port 80. Additionally, the determination of which requests are directed to port 80 is not in response to the incoming IP Datagram originating from a socks client.

Furthermore, there is not so much as a suggestion in any of the references to modify the references to include such features. That is, there is no teaching or suggestion in Cohen, Kayashima and Jade that a problem exists for which determining whether the incoming IP Datagram originates from a socks client or from a socks server; and in response to the incoming IP Datagram originating from a socks client: terminating the TCP connection and the socks connection; identifying the socks connection in a table; identifying the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol; and determining whether said application level protocol is HTTP or not, is a solution. To the contrary, Cohen teaches performing redirection of connections from an origin server to a proxy server. Kayashima teaches connectionless communication in a network communication system. Jade teaches a firewall that isolates computer and network resources inside the firewall from networks, computers and computer applications outside the firewall. None of the reference even recognizes a need to determining whether the incoming IP Datagram originates from a socks client or from a socks server, as recited in claim 1.

One of ordinary skill in the art, being presented only with Cohen, Kayashima and Jade, and without having a prior knowledge of Applicant's claimed invention, would not have found it obvious to combine and modify Cohen, Kayashima and Jade to arrive at Applicant's claimed invention. To the contrary, even if one were somehow motivated to combine Cohen, Kayashima and Jade, and it were somehow possible to combine the systems, the result would not be the invention as recited in claim 1. The result would simply identify the element of an IP request and establish a connection from a client to a proxy server using sockets that are defined in a table. The resulting system still would not determine whether the incoming IP Datagram originates from a socks client or from a socks server and in response to the incoming IP Datagram originating from a socks client: terminating the TCP connection and the socks connection; identifying the socks connection in a table; identifying the application level protocol associated with said socks connection referring to said table, said table comprising for each socks connection an application level protocol; and determining whether said application level protocol is HTTP or not.

Thus, Cohen, Kayashima and Jade, taken alone or in combination, fail to teach or suggest all of the features in independent claim 1. At least by virtue of their dependency on claim 1, the specific features of claims 2 and 4 are not taught or suggested by Cohen, Kayashima and Jade, either alone or in combination. Accordingly, Applicant respectively requests withdrawal of the rejection of claims 1, 2 and 4 under 35 U.S.C. § 103(a).

Moreover, in addition to their dependency from independent claims 1, 10 and 11 respectively, the specific features recited in dependent claims 2, 4-9 and 12, 13 and 15-20 are not taught or suggested by Cohen, Kayashima and Jade, taken alone or in combination. For example, with regard to claim 2 and similarly claims 12 and 13, Cohen, Kayashima and Jade, taken alone or in combination, fail to teach or suggest in response to the HTTP data requested by the IP Datagram not residing in the local cache within the network device: identifying the outbound socks connection associated with the socks connection referring to the table, said table comprising for each socks connection an outbound socks connection; building an outgoing IP Datagram with information comprised in the incoming IP Datagram; and sending said outgoing IP Datagram on the outbound socks connection. The Officc Action alleges that this feature is taught by Cohen at column 5, lines 10-30, which reads as follows, and column 7, lines 10-50, shown above.

> The problems associated with the prior art techniques for transparent proxy caching are eliminated by the present invention. In accordance with the present invention, a switching entity, such as the L4 switch (referred to hereinafter as a proxy redirector), through which the packets flow, is provided with the functionalities at the IP level necessary to transform the complete URL in each GET request transmitted by a client to an appropriate absolute URL. Specifically, the IP address found in the destination field in the IP header of the packet(s) from the client containing the GET request are added as a prefix by the proxy redirector to the complete URL in the GET request. As a result, the complete URL in the GET request is modified to form an absolute URL which, when received by the proxy cache, is directly used to determine if the requested object is stored in the cache and, if not, to establish a separate TCP connection to the origin server. The GET request received by the proxy is thus equivalent to what it would expect to receive if it were operating in the non-transparent mode. Advantageously, if a persistent connection is established, each subsequent GET request has the same IP address prefix determined by the initial DNS look-up by the client.

(Column 5, lines 10-30)

In these sections, Cohen is describing establishing a connection from a client to a proxy server instead of to the intended origin server. If the information requested by the client is not found on the proxy server, a separate TCP connection is established from the proxy server to the origin server. Thus, Cohen does not teach or suggest, in response to the HTTP data requested by the IP Datagram not residing in the local cache within the network device<u>; identifying the outbound socks connection associated with the socks connection referring to the table</u>, said table comprising for each socks connection an outbound socks connection. Cohen merely establishes a new TCP connection from the proxy server to the origin server. Kayashima and Jade do not make up for the deficiencies of Cohen.

Additionally, with regard to claims 4 and similarly claim 15, Cohen, Kayashima and Jade, taken alone or in combination, fail to teach or suggest identifying the outbound socks connection associated with the socks connection referring to the table comprises the further steps of: defining an outbound socks connection between the network device and the destination device of the incoming IP Datagram; and associating in the table said outbound socks connection with the socks connection of the incoming IP Datagram. As discussed above, Cohen teaches establishing a connection from a client to a proxy server instead of to the intended for an origin server and, if the information requested by the client is not found on the proxy server, a separate TCP connection is established from the proxy server to the origin server. Thus, there would be no need for Cohen to define an outbound socks connection between a network device and a destination device of the incoming IP Datagram; and associating in the table said outbound socks connection with the socks connection of the incoming IP Datagram. Kayashima and Jade do not make up for the deficiencies of Cohen.

Thus, in addition to being dependent on independent claims 1, 10 and 11, the specific features of dependent claims 2 and 4 are also distinguishable over the combination of Cohen, Kayashima and Jade by virtue of the specific features recited in these claims. Accordingly, Applicant respectfully requests withdrawal of the rejection of dependent claims 2 and 4 under 35 U.S.C. § 103 (a).

II. <u>35 U.S.C. § 103, Alleged Obviousness, Claim 3</u>

The Office Action rejects claim 3 under 35 U.S.C. § 103(a) as being unpatentable over Cohen et al. (U.S. Patent No. 6,389,462 B1) in view of Kayashima (U.S. Patent No. 6,195,366 B1) and further in view of Jade et al. (U.S. Patent No. 5,944,823) in further view of Bhagwat et al. (U.S. Patent No. 5,941,988). This rejection is respectfully traversed.

Claim 3 and similarly claim 14 are dependent on independent claims 1 and 10 and, thus, these claims distinguish over the combination of Cohen, Kayashima and Jade for at least the reasons noted above with regards to claims 1 and 10. Moreover, Bhagwat does not provide for the deficiencies of Cohen, Kayasbima and Jade, and, thus, any alleged combination of Cohen, Kayashima, Jade and Bhagwat would not be sufficient to reject independent claims 1 and 10 or claims 3 and 14 by virtue of their dependency.

Moreover, the Office Action may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Such reliance is an impermissible use of hindsight with the benefit of Applicant's disclosure. *Id.* Therefore, absent some teaching, suggestion, or incentive in the prior art, Cohen, Kayashima, Jade and Bhagwat cannot be properly combined to form the claimed invention. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed combination, the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of Applicant's disclosure as a model for the needed changes.

In view of the above, Cohen, Kayashima, Jade and Bhagwat, taken either alone or in combination, fail to teach or suggest the specific features recited in independent claim 1, from which claim 3 depends. Accordingly, Applicant respectfully requests withdrawal of the rejection of claim 3 under 35 U.S.C. § 103.

III. Claims 5-20

The Office Action fails to provide a section where the specific features of claims 5-20 are taught or suggested by the combination of Cohen, Kayashima, Jade and Bhagwhat, but, rather, states:

> Claims 5-20, do not teach or define any new limitation which is not inherent in the above mentioned claims 1-4, and therefore are rejected for similar reasons.

Applicant respectfully traverses this statement. Claims 10-15 have been addressed above with respect to claims 1-4. Applicant respectfully submits that claims 5-9 and 16-20 do, in fact, define limitations which are not inherent in claims 1-4. For example, claims 5 and 16 recite <u>in response to the incoming IP Datagram not originating from a socks server</u>: terminating the TCP connection and the socks connection; identifying the socks connection in the table; identifying the application level protocol associated with said socks connection referring to said table; and determining whether said application level protocol is HTTP; and in response to said application level protocol being HTTP: caching HTTP data comprised in incoming IP Datagram in the local cache of the network device; identifying the inbound socks connection associated with the socks connection referring to the table, said table comprising for each socks connection an inbound socks connection; building an outgoing IP Datagram with information comprised in the incoming IP Datagram; and sending said outgoing IP Datagram on the inbound socks connection. Applicant respectfully submits that at least the emphasized features are not found nor are they inherent from the claim language of claims 1-4. Furthermore, as discussed above, the combination of Cohen, Kayashima and Jade fails to teach or suggest in response to the incoming IP Datagram originating from a socks server, thus, the combination fails to teach or suggest in response to the incoming IP Datagram not originating from a socks server.

As an additional example, claims 7 and 18 recite wherein the step of determining whether the IP Datagram is originated by a socks client or a socks server comprises the step of: determining if the value of the Destination Port field comprised in the IP Datagram is equal to the value of a destination port on a socks server or if the value of the Source Port field comprised in the IP Datagram is equal to the value of a source port on a socks server. These features do not appear and are not inherent from claims 1-4. Moreover, none of the references teaches or suggests equating the value of the Destination Port field comprised in the IP Datagram to the value of a destination port on a socks server or equating the value of the Source Port field comprised in the IP Datagram to the value of a source port on a socks server.

As a further example, claims 8 and 19 recite wherein said table is dynamic and comprises for each socks connection: an identification of the inbound socks connection; an identification of the associated outbound connection; and an identification of the application level protocol used in IP Datagrams using said socks connection. These features do not reside or is inherent from claims 1-4. None of the references teaches or suggests a table that is dynamic and comprises an identification of the inbound socks connection; an identification of the associated outbound connection; and an identification of the application level protocol used in IP Datagrams using said socks connection.

As still a further example, claims 9 and 20 recite wherein said table comprises: for identifying each inbound socks connection: an inbound source device address identifying the source device of the inbound socks connection; an inbound source port address identifying the source port of the inbound socks connection; an inbound destination device address identifying the destination device of the inbound socks connection; and an inbound destination port address identifying the destination port of the inbound socks connection; and for identifying each outbound socks connection: an outbound source device address identifying the source device of the outbound socks connection; an outbound source application address identifying the source port of the outbound socks connection; an outbound destination device address identifying the destination device of the outbound socks connection; and an outbound destination application address identifying the destination port of the outbound socks connection. These features do not appear in and are not inherent from, claims 1-4. Moreover, none of the references teaches or suggests a table comprised of all the features listed in these claims.

Thus, the Office Action fails to address all of the features of the presently claimed invention. Moreover, Cohen, Kayashima, Jade and Bhagwat, taken alone or in combination, fail to teach or suggest all of the features in independent claims 1, 10 and 11. At least by virtue of their dependency on claims 1, 10 and 11, the specific features of claims 5-9, 12 and 13-20 are not taught or suggested by Cohen, Kayashima, Jade and Bhagwat, either alone or in combination. Accordingly, Applicant respectively requests withdrawal of the rejection of claims 5-20 under 35 U.S.C. § 103(a).

IV. Conclusion

It is respectfully urged that the subject application is patentable over the prior art of record and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: December 14, 2004

Francis Lammes

Francis Lammes
Reg. No. 55,353
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicant